Manual de Implementação Streamlit com AWS Bedrock

Sumário

1. Configuração do AWS CLI com SSO

Para configurar o AWS CLI com Single Sign-On (SSO), siga estas etapas:

1. Certifique-se de que o AWS CLI está instalado na sua máquina. Caso ainda não tenha instalado, baixe aqui.
2. Abra o terminal e execute o comando:

```bash
aws configure sso
```

3. Você será solicitado a fornecer as seguintes informações:
    - **SSO start URL**: URL de início fornecida pelo administrador (exemplo: https://seudominio.awsapps.com/start)
    - **SSO Region**: Região onde o SSO está configurado (exemplo: us-east-1)

- **Default CLI Profile Name**: Nome do perfil para esse conjunto de credenciais (exemplo: seu-perfil-nome)

4. Após inserir essas informações, uma página do navegador será aberta para você fazer login.
5. Após o login bem-sucedido, você verá uma lista de contas AWS disponíveis. Selecione a conta com a qual deseja trabalhar.
6. Em seguida, selecione o perfil de permissão (role) que deseja usar.
7. Configure as opções padrão (região padrão, formato de saída, etc.)
8. A configuração está completa. Para testar, execute:

```bash
aws sts get-caller-identity --profile seu-perfil-nome
```

Este comando deve retornar suas informações de identidade na AWS, confirmando que a configuração foi bem-sucedida.

2. Configuração Local com Access Key e Secret Key

Se preferir usar credenciais de longo prazo (não recomendado para ambientes de produção), siga as etapas abaixo:

1. No Console AWS, navegue até **IAM > Usuários** e selecione seu usuário.
2. Vá para a aba **Credenciais de segurança**.
3. Em **Chaves de acesso**, clique em **Criar chave de acesso**.
4. Após a criação, você verá sua **Access Key ID** e **Secret Access Key**. Anote-as com segurança (esta é a única vez que você verá a Secret Access Key).
5. No terminal, execute:

```bash
aws configure --profile seu-perfil-nome
```

6. Insira as seguintes informações quando solicitado:
   - AWS Access Key ID
   - AWS Secret Access Key
   - Default region name (exemplo: us-east-1)
   - Default output format (exemplo: json)
7. Para testar a configuração, execute:

bash

```
aws sts get-caller-identity --profile seu-perfil-nome
```

O resultado deve mostrar suas informações de identidade na AWS.

3. Obtendo o Código Base

Clone o repositório base para sua máquina local:

bash

```
git clone https://github.com/marcos-px/streamlit-base.git
cd streamlit-base
```

Isso criará uma pasta `streamlit-base` com todo o código necessário.

4. Personalizando o Código

Configuração do Perfil AWS

1. Abra o arquivo `streamlit/.env` e atualize com o nome do perfil AWS que você criou:

```
AWS_PROFILE="seu-perfil-nome"
```

2. Também atualize o arquivo `streamlit/app.py`, localizando a linha:

python

```
PROFILE_NAME = os.environ.get("AWS_PROFILE", "edn174")
```

Substitua "edn174" pelo nome do seu perfil.

3. Atualize também em outros locais onde o perfil é mencionado, como na função `get_boto3_client`:

python

```
def get_boto3_client(service_name, region_name='us-east-1', profile_name='seu-perfil-nome'):
```

4. Atualize o ARN do profile de inferência no arquivo `app.py`: Apenas se for usar um modelo diferente

python

```
INFERENCE_PROFILE_ARN = "arn:aws:bedrock:us-east-1:YourAccountId:inference-profile/modelo-desej
```

Personalização do Prompt

1. Abra o arquivo `streamlit/functions.py` e localize a função `generate_chat_prompt`.
2. Modifique o `system_prompt` para refletir o comportamento desejado para seu assistente:

python

```python
def generate_chat_prompt(user_message, conversation_history=None, context=""):
    """
    Gera um prompt de chat completo com histórico de conversa e contexto opcional.
    """
    system_prompt = """
    Você é um assistente virtual especializado em [SUA ESPECIALIDADE].
    [INSTRUÇÕES ESPECÍFICAS PARA O ASSISTENTE]
    [COMPORTAMENTO ESPERADO]
    [QUALQUER OUTRA INSTRUÇÃO RELEVANTE]
    """
    # Resto da função permanece igual
```

Ajuste dos Parâmetros do Modelo

No arquivo `streamlit/app.py`, localize a função `query_bedrock` e ajuste os parâmetros do modelo conforme necessário:

python

```python
def query_bedrock(message, session_id="", model_params=None, context=""):
    if model_params is None:
        model_params = {
            "temperature": 0.7,  # Ajuste conforme necessário (0.0 a 1.0)
            "top_p": 0.9,        # Ajuste conforme necessário (0.0 a 1.0)
            "top_k": 250,        # Número de tokens a considerar
            "max_tokens": 1000,  # Tamanho máximo da resposta
            "response_format": {"type": "text"}
        }
    # Resto da função permanece igual
```

Também atualize os parâmetros na função `invoke_bedrock_model` em `functions.py`:

```python
def invoke_bedrock_model(prompt, inference_profile_arn, model_params=None):
    if model_params is None:
        model_params = {
            "temperature": 0.7,  # Ajuste conforme necessário (0.0 a 1.0)
            "top_p": 0.9,        # Ajuste conforme necessário (0.0 a 1.0)
            "top_k": 250,        # Número de tokens a considerar
            "max_tokens": 1000   # Tamanho máximo da resposta
        }
    # Resto da função permanece igual
```

Alteração de Usuário e Senha

No arquivo `streamlit/app.py`, localize a função `check_password` e altere o usuário e senha:

```python
def password_entered():
    """Checks whether a password entered by the user is correct."""
    print(f"DEBUG LOGIN: Tentativa de login - Usuário: {st.session_state['username']}', Senha:

    if hmac.compare_digest(st.session_state["username"].strip(), "seu-usuario") and \
    hmac.compare_digest(st.session_state["password"].strip(), "sua-senha"):
        print("DEBUG LOGIN: Autenticação bem-sucedida")
        # Resto da função permanece igual
```

Substitua "seu-usuario" e "sua-senha" pelas credenciais desejadas.

Personalizando a Aparência

No arquivo `streamlit/app.py`, você pode alterar o título da página e o ícone:

```python
st.set_page_config(
   page_title="NOME DA SUA APLICAÇÃO",
   page_icon="logo.jpeg",
   layout="wide",
   initial_sidebar_state="expanded"
)
```

Substitua "NOME DA SUA APLICAÇÃO" pelo título desejado e, se quiser, coloque sua própria imagem para o logo no lugar de "logo.jpeg".

Desabilitando Recursos

Para desabilitar o RAG (Retrieval Augmented Generation):

1. No arquivo `streamlit/app.py`, localize a seção relacionada ao RAG:

```python
use_rag = st.checkbox("Usar Contexto Adicional (RAG)", value=st.session_state.use_rag)
st.session_state.use_rag = use_rag

if use_rag:
    # código do RAG...
```

2. Para desabilitar, comente ou remova esta seção, ou simplesmente altere o valor padrão para `False`:

```python
use_rag = st.checkbox("Usar Contexto Adicional (RAG)", value=False)  # Alterado para False
st.session_state.use_rag = use_rag

# Comentar ou remover toda esta seção se quiser desativar completamente
# if use_rag:
#     # código do RAG...
```

Para desabilitar o anexo de arquivos:

1. No arquivo `streamlit/app.py`, localize a seção de upload de arquivos:

```python
with col2:
    file_to_send = st.file_uploader("Anexar arquivo", type=["pdf", "txt", "csv", "doc", "docx",
                                    key="file_to_send", label_visibility="collapsed")
    st.markdown('<div class="attach-icon" title="Anexar arquivo"><i class="fas fa-paperclip"></
```

2. Comente ou remova esta seção e ajuste o layout das colunas:

python

```python
# Remova col2 para transformar de 3 colunas para 2
col1, col3 = st.columns([5, 1])
# ou para usar toda a largura
# col1 = st.columns([1])[0]

with col1:
    st.text_area("Mensagem", placeholder="Digite sua mensagem aqui...", key="user_input",
        height=70, label_visibility="collapsed")

# with col2:  # Comentado para desabilitar anexos
#    file_to_send = st.file_uploader("Anexar arquivo", type=["pdf", "txt", "csv", "doc", "docx"
#                                    key="file_to_send", label_visibility="collapsed")
#    st.markdown('<div class="attach-icon" title="Anexar arquivo"><i class="fas fa-paperclip"><

with col3:
    if st.button("Enviar", key="send_button", use_container_width=True):
        if st.session_state.user_input and st.session_state.user_input.strip():
            handle_message()
```

3. Também remova ou comente a lógica de processamento de arquivos anexados na função `handle_message`.

5. Testando a Aplicação Localmente

1. Crie um ambiente virtual Python e ative-o:

bash

```bash
python -m venv .venv
source .venv/bin/activate # No Windows: .venv\Scripts\activate
```

2. Instale as dependências:

bash

```bash
pip install -r requirements.txt
```

3. Execute a aplicação:

bash

```bash
cd streamlit
streamlit run app.py
```

4. A aplicação será aberta em seu navegador, geralmente em http://localhost:8501.
5. Teste todas as funcionalidades para garantir que tudo está funcionando como esperado.

6. Preparando para Implantação no EC2

Para implantar na AWS EC2, você precisa remover as configurações específicas do perfil local:

1. Crie uma nova branch para a versão de produção:

bash

```
git checkout -b production
```

2. No arquivo `streamlit/app.py`, remova as referências ao perfil AWS local:

python

```
# Altere
PROFILE_NAME = os.environ.get("AWS_PROFILE", "seu-perfil-nome")

# Para
PROFILE_NAME = os.environ.get("AWS_PROFILE", "")
```

3. Na função `get_boto3_client`, ajuste para usar as credenciais da instância EC2:

python

```python
def get_boto3_client(service_name, region_name='us-east-1', profile_name=''):
    """
    Retorna um cliente do serviço AWS especificado.

    Na EC2, usará automaticamente o perfil de instância.
    """
    try:
        # Se um perfil for fornecido, tenta usá-lo
        if profile_name:
            session = boto3.Session(profile_name=profile_name, region_name=region_name)
            client = session.client(service_name)
            return client
        # Caso contrário, usa as credenciais da instância EC2
        else:
            session = boto3.Session(region_name=region_name)
            client = session.client(service_name)
            return client
    except Exception as e:
        print(f"ERRO: Falha ao criar cliente boto3:{str(e)}")
        return None
```

4. Certifique-se de que seu arquivo `.gitignore` inclua os arquivos que não devem ir para o repositório:

```
.venv/
.env
*.pyc
__pycache__/
```

5. Depois faça commit das alterações:

bash

```bash
git add .
git commit -m "Preparado para implantação na EC2"
```

7. Criando e Configurando uma Instância EC2

Criando um Grupo de Segurança

1. No console AWS, vá para **EC2 > Segurança > Grupos de segurança** e clique em **Criar grupo de segurança**.
2. Preencha os detalhes:

- **Nome**: streamlit-app-sg
- **Descrição**: Grupo de segurança para aplicação Streamlit
- **VPC**: Selecione sua VPC

3. Adicione as regras de entrada:
   - Para SSH (para seu acesso):
     - **Tipo**: SSH
     - **Protocolo**: TCP
     - **Intervalo de portas**: 22
     - **Origem**: Seu IP (ex: 123.123.123.123/32)
     - **Descrição**: SSH apenas do IP do desenvolvedor
   - Para Streamlit:
     - **Tipo**: TCP personalizado
     - **Protocolo**: TCP
     - **Intervalo de portas**: 8501
     - **Origem**: 0.0.0.0/0 (qualquer lugar) ou restrinja se necessário
     - **Descrição**: Acesso à porta Streamlit
4. Clique em **Criar grupo de segurança**.

Lançando a Instância EC2

1. No console AWS, vá para **EC2 > Instâncias** e clique em **Executar instâncias**.
2. Configure a instância:
   - **Nome**: streamlit-app
   - **AMI**: Procure pela AMI `ami-04b33c5c5e7fdd712`
   - **Tipo de instância**: t2.micro ou maior, dependendo das necessidades
   - **Par de chaves**: Crie um novo par ou selecione um existente
   - **Configurações de rede**: Selecione uma VPC e sub-rede
   - **Grupo de segurança**: Selecione o grupo criado anteriormente (streamlit-app-sg)
   - **Configuração de armazenamento**: 8GB ou mais, dependendo das necessidades
3. Clique em **Executar instância**.
4. Aguarde a instância iniciar (status "running").
5. Anote o **IP público** da instância. Você usará esse IP para se conectar via SSH e para acessar a aplicação.

8. Enviando Código para Repositório Pessoal

1. Crie um novo repositório no GitHub (sem inicializar com README ou arquivo).
2. Configure seu repositório local para apontar para o novo repositório:

bash

```bash
# Remova o remote atual
git remote remove origin

# Adicione seu novo repositório
git remote add origin https://github.com/seu-usuario/seu-repo.git
```

3. Envie seu código para o GitHub:

bash

```bash
# Envie a branch principal
git push -u origin main

# Envie a branch de produção, se criada
git push -u origin production
```

9. Criando Chave SSH na Instância EC2

1. Conecte-se à instância EC2 usando o par de chaves criado anteriormente:

bash

```bash
ssh -i caminho/para/sua-chave.pem ec2-user@seu-ip-publico
```

2. Dentro da instância, gere um novo par de chaves SSH:

bash

```bash
ssh-keygen -t ed25519 -C "seu-email@exemplo.com"
```

3. Pressione Enter para aceitar o local padrão e deixe a senha em branco (pressione Enter duas vezes).
4. Exiba a chave pública:

bash

```bash
cat ~/.ssh/id_ed25519.pub
```

5. Copie a saída completa da chave.
6. No GitHub, vá para **Configurações > Chaves SSH e GPG > Nova chave SSH**.
7. Forneça um título (ex: "EC2 Streamlit App") e cole a chave pública.

8. Clique em **Adicionar chave SSH**.
9. Teste a conexão SSH com o GitHub:

```bash
ssh -T git@github.com
```

Você deve ver uma mensagem de sucesso.

10. Implantando o Código na Instância EC2

1. Dentro da instância EC2, clone seu repositório:

```bash
git clone git@github.com:seu-usuario/seu-repo.git
cd seu-repo
```

2. Se você criou uma branch de produção separada, mude para ela:

```bash
git checkout production
```

3. Configure o ambiente Python:

```bash
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

11. Executando Streamlit com tmux

O tmux permite manter a aplicação em execução mesmo após você se desconectar da sessão SSH.

1. Instale o tmux (se ainda não estiver instalado):

```bash
sudo yum install -y tmux
```

2. Inicie uma nova sessão tmux:

```bash
tmux new -s streamlit
```

3. Ative o ambiente virtual e inicie a aplicação:

```bash
cd seu-repo/streamlit
source ../.venv/bin/activate
streamlit run app.py
```

4. Desacople da sessão tmux pressionando `Ctrl+b` seguido de `d`. Isso manterá a aplicação em execução.
5. Você pode reconectar à sessão posteriormente com:

```bash
tmux attach -t streamlit
```

12. Acessando a Aplicação

1. Agora você pode acessar sua aplicação através do navegador em:

```
http://seu-ip-publico:8501
```

Nota importante: Use sempre `http://` e não `https://`, já que o uso de https sem a configuração adequada resultará em uma tela de carregamento infinita.

2. Se você precisar interromper a aplicação, reconecte-se à sessão tmux e pressione `Ctrl+C`.
3. Para sair e encerrar completamente a sessão tmux, use:

```bash
exit
```

Considerações Finais

- Certifique-se de que o perfil IAM associado à instância EC2 tem as permissões necessárias para acessar o Amazon Bedrock e outros serviços AWS utilizados.
- Para ambientes de produção, considere implementar HTTPS adequadamente.
- Monitore o uso da sua instância EC2 e do Amazon Bedrock para controlar custos.
- Implemente um processo de CI/CD para automaticar atualizações futuras da aplicação.
- Faça backup regular dos dados importantes da sua aplicação.

Parabéns! Você agora tem uma aplicação Streamlit funcionando com o Amazon Bedrock!